Gradient Descent Optimization

Nipun Batra

IIT Gandhinagar

July 30, 2025

Outline

Gradient: Direction of Steepest Ascent

- **Gradient** denotes the direction of steepest *ascent*

Gradient: Direction of Steepest Ascent

- **Gradient** denotes the direction of steepest *ascent*
- Direction in which there is maximum *increase* in $f(x, y)$

Gradient: Direction of Steepest Ascent

- **Gradient** denotes the direction of steepest *ascent*
- Direction in which there is maximum *increase* in $f(x, y)$
- For *descent*, we move in the opposite direction: $-\nabla f$

Gradient: Direction of Steepest Ascent

- **Gradient** denotes the direction of steepest *ascent*
- Direction in which there is maximum *increase* in $f(x, y)$
- For *descent*, we move in the opposite direction: $-\nabla f$

Gradient: Direction of Steepest Ascent

- **Gradient** denotes the direction of steepest *ascent*
- Direction in which there is maximum *increase* in $f(x, y)$
- For *descent*, we move in the opposite direction: $-\nabla f$

Example: $f(x, y) = x^2 + y^2$

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

Pop Quiz: Gradient Direction

Quick Quiz 1

If we want to minimize a function, which direction should we move?

a) Direction of the gradient ∇f

Answer: b) Opposite to gradient $-\nabla f$ for steepest descent!

Pop Quiz: Gradient Direction

Quick Quiz 1

If we want to minimize a function, which direction should we move?

a) Direction of the gradient ∇f
b) Opposite to the gradient $-\nabla f$

Answer: b) Opposite to gradient $-\nabla f$ for steepest descent!

Pop Quiz: Gradient Direction

Quick Quiz 1

If we want to minimize a function, which direction should we move?

a) Direction of the gradient ∇f
b) Opposite to the gradient $-\nabla f$
c) Perpendicular to the gradient

Answer: b) Opposite to gradient $-\nabla f$ for steepest descent!

Batch vs Stochastic Gradient Descent

Batch vs Stochastic Gradient Descent

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration
- More memory required

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration
- More memory required

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration
- More memory required

Stochastic Gradient Descent

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration
- More memory required

Stochastic Gradient Descent

- Use **one** data point

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration
- More memory required

Stochastic Gradient Descent

- Use **one** data point
- Noisier convergence path

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration
- More memory required

Stochastic Gradient Descent

- Use **one** data point
- Noisier convergence path
- Faster per iteration

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration
- More memory required

Stochastic Gradient Descent

- Use **one** data point
- Noisier convergence path
- Faster per iteration
- Less memory required

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration
- More memory required

Stochastic Gradient Descent

- Use **one** data point
- Noisier convergence path
- Faster per iteration
- Less memory required

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration
- More memory required

Stochastic Gradient Descent

- Use **one** data point
- Noisier convergence path
- Faster per iteration
- Less memory required

- **SGD:** Update parameters after seeing *each* point

Batch vs Stochastic Gradient Descent

Batch Gradient Descent

- Use **all** training data
- Smooth convergence
- Slower per iteration
- More memory required

Stochastic Gradient Descent

- Use **one** data point
- Noisier convergence path
- Faster per iteration
- Less memory required

- **SGD:** Update parameters after seeing *each* point
- **Key benefit:** Computes gradient over one example $\rightarrow$ less time per update

Pop Quiz: SGD vs Batch GD

Quick Quiz 2

For a dataset with 1 million samples, which converges faster per iteration?

a) Batch Gradient Descent (uses all 1M samples)

Answer: b) SGD is much faster per iteration, but needs more iterations!

Pop Quiz: SGD vs Batch GD

Quick Quiz 2

For a dataset with 1 million samples, which converges faster per iteration?

a) Batch Gradient Descent (uses all 1M samples)
b) Stochastic Gradient Descent (uses 1 sample)

Answer: b) SGD is much faster per iteration, but needs more iterations!

Pop Quiz: SGD vs Batch GD

Quick Quiz 2

For a dataset with 1 million samples, which converges faster per iteration?

a) Batch Gradient Descent (uses all 1M samples)
b) Stochastic Gradient Descent (uses 1 sample)
c) They're the same speed

Answer: b) SGD is much faster per iteration, but needs more iterations!

Gradient Descent: Smart Implementation

Standard Form

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Gradient Descent: Smart Implementation

Standard Form

$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$

Gradient Descent: Smart Implementation

Standard Form

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Optimized Form

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top \mathbf{X}\boldsymbol{\theta} + \alpha \mathbf{X}^\top \mathbf{y}$$

Gradient Descent: Smart Implementation

Standard Form

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Optimized Form

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top \mathbf{X}\boldsymbol{\theta} + \alpha \mathbf{X}^\top \mathbf{y}$$

Gradient Descent: Smart Implementation

Standard Form

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$$

Optimized Form

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top \mathbf{X}\boldsymbol{\theta} + \alpha \mathbf{X}^\top \mathbf{y}$$

Key insight: $\mathbf{X}^\top \mathbf{X}$ and $\mathbf{X}^\top \mathbf{y}$ can be *precomputed once!*

Precomputation Costs

One-time computations:

- Computing $\mathbf{X}^\top \mathbf{y}$: $\mathcal{O}(dn)$

Precomputation Costs

One-time computations:

- Computing $\mathbf{X}^\top \mathbf{y}$: $\mathcal{O}(dn)$

Precomputation Costs

One-time computations:

- Computing $\mathbf{X}^\top \mathbf{y}$: $\mathcal{O}(dn)$
- Computing $\mathbf{X}^\top \mathbf{X}$: $\mathcal{O}(d^2 n)$

Precomputation Costs

One-time computations:

- Computing $\mathbf{X}^\top \mathbf{y}$: $\mathcal{O}(dn)$
- Computing $\mathbf{X}^\top \mathbf{X}$: $\mathcal{O}(d^2 n)$

Precomputation Costs

One-time computations:

- Computing $\mathbf{X}^\top \mathbf{y}$: $\mathcal{O}(dn)$
- Computing $\mathbf{X}^\top \mathbf{X}$: $\mathcal{O}(d^2 n)$
- Scaling by α: $\mathcal{O}(d^2)$ for $\alpha \mathbf{X}^\top \mathbf{X}$

Precomputation Costs

One-time computations:

- Computing $\mathbf{X}^\top \mathbf{y}$: $\mathcal{O}(dn)$
- Computing $\mathbf{X}^\top \mathbf{X}$: $\mathcal{O}(d^2 n)$
- Scaling by α: $\mathcal{O}(d^2)$ for $\alpha \mathbf{X}^\top \mathbf{X}$

Precomputation Costs

One-time computations:

- Computing $\mathbf{X}^\top \mathbf{y}$: $\mathcal{O}(dn)$
- Computing $\mathbf{X}^\top \mathbf{X}$: $\mathcal{O}(d^2 n)$
- Scaling by α: $\mathcal{O}(d^2)$ for $\alpha \mathbf{X}^\top \mathbf{X}$

Total Precomputation Cost

$\mathcal{O}(d^2 n) + \mathcal{O}(dn) + \mathcal{O}(d^2) = \mathcal{O}(d^2 n)$

Per-Iteration Costs (Optimized)

For each of t iterations:

- Matrix-vector multiplication: $\alpha \mathbf{X}^\top \mathbf{X} \cdot \boldsymbol{\theta}$

Per-Iteration Costs (Optimized)

For each of t iterations:

- Matrix-vector multiplication: $\alpha \mathbf{X}^\top \mathbf{X} \cdot \boldsymbol{\theta}$
  - $(d \times d)$ matrix $\times$ $(d \times 1)$ vector = $\mathcal{O}(d^2)$

Per-Iteration Costs (Optimized)

For each of t iterations:

- Matrix-vector multiplication: $\alpha \mathbf{X}^\top \mathbf{X} \cdot \boldsymbol{\theta}$
  - $(d \times d)$ matrix $\times$ $(d \times 1)$ vector = $\mathcal{O}(d^2)$

Per-Iteration Costs (Optimized)

For each of t iterations:

- Matrix-vector multiplication: $\alpha \mathbf{X}^\top \mathbf{X} \cdot \boldsymbol{\theta}$
  - $(d \times d)$ matrix $\times$ $(d \times 1)$ vector = $\mathcal{O}(d^2)$
- Vector operations: Addition/subtraction in $\mathcal{O}(d)$

Per-Iteration Costs (Optimized)

For each of t iterations:

- Matrix-vector multiplication: $\alpha\mathbf{X}^\top\mathbf{X}\cdot\boldsymbol{\theta}$
  - $(d \times d)$ matrix $\times$ $(d \times 1)$ vector = $\mathcal{O}(d^2)$
- Vector operations: Addition/subtraction in $\mathcal{O}(d)$

Per-Iteration Costs (Optimized)

For each of t iterations:

- Matrix-vector multiplication: $\alpha\mathbf{X}^\top\mathbf{X}\cdot\boldsymbol{\theta}$
  - $(d \times d)$ matrix $\times$ $(d \times 1)$ vector = $\mathcal{O}(d^2)$
- Vector operations: Addition/subtraction in $\mathcal{O}(d)$

Total Complexity (Optimized)

Precomputation + t iterations:
$\mathcal{O}(d^2 n) + \mathcal{O}(td^2) = \mathcal{O}((n+t)d^2)$

Naive Implementation Complexity

Without precomputation: $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$
Per iteration costs:

- Computing $\mathbf{X}\boldsymbol{\theta}$: $\mathcal{O}(nd)$

Naive Implementation Complexity

Without precomputation: $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$
Per iteration costs:

- Computing $\mathbf{X}\boldsymbol{\theta}$: $\mathcal{O}(nd)$
- Computing $\mathbf{X}\boldsymbol{\theta} - \mathbf{y}$: $\mathcal{O}(n)$

Naive Implementation Complexity

Without precomputation: $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$
Per iteration costs:

- Computing $\mathbf{X}\boldsymbol{\theta}$: $\mathcal{O}(nd)$
- Computing $\mathbf{X}\boldsymbol{\theta} - \mathbf{y}$: $\mathcal{O}(n)$

Naive Implementation Complexity

Without precomputation: $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$
Per iteration costs:

- Computing $\mathbf{X}\boldsymbol{\theta}$: $\mathcal{O}(nd)$
- Computing $\mathbf{X}\boldsymbol{\theta} - \mathbf{y}$: $\mathcal{O}(n)$
- Computing $\mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$: $\mathcal{O}(nd)$

Naive Implementation Complexity

Without precomputation: $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$
Per iteration costs:

- Computing $\mathbf{X}\boldsymbol{\theta}$: $\mathcal{O}(nd)$
- Computing $\mathbf{X}\boldsymbol{\theta} - \mathbf{y}$: $\mathcal{O}(n)$
- Computing $\mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$: $\mathcal{O}(nd)$
- Final update: $\mathcal{O}(d)$

Naive Implementation Complexity

Without precomputation: $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$
Per iteration costs:

- Computing $\mathbf{X}\boldsymbol{\theta}$: $\mathcal{O}(nd)$
- Computing $\mathbf{X}\boldsymbol{\theta} - \mathbf{y}$: $\mathcal{O}(n)$
- Computing $\mathbf{X}^\top(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$: $\mathcal{O}(nd)$
- Final update: $\mathcal{O}(d)$

Naive Implementation Complexity

Without precomputation: $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha \mathbf{X}^\top (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$
Per iteration costs:

- Computing $\mathbf{X}\boldsymbol{\theta}$: $\mathcal{O}(nd)$
- Computing $\mathbf{X}\boldsymbol{\theta} - \mathbf{y}$: $\mathcal{O}(n)$
- Computing $\mathbf{X}^\top(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})$: $\mathcal{O}(nd)$
- Final update: $\mathcal{O}(d)$

Total Complexity (Naive)

$t \times \mathcal{O}(nd) = \mathcal{O}(ndt)$

Pop Quiz: Complexity Comparison

Quick Quiz 3

For $d = 100$, $n = 10000$, $t = 1000$, which implementation is more efficient?

a) Optimized: $\mathcal{O}((n + t)d^2)$ approach

Answer: a) Optimized has better asymptotic complexity when $n + t < nd$!

Pop Quiz: Complexity Comparison

Quick Quiz 3

For $d = 100$, $n = 10000$, $t = 1000$, which implementation is more efficient?

a) Optimized: $\mathcal{O}((n+t)d^2)$ approach
b) Naive: $\mathcal{O}(ndt)$ approach

Answer: a) Optimized has better asymptotic complexity when $n + t < nd$!

Pop Quiz: Complexity Comparison

Quick Quiz 3

For $d = 100$, $n = 10000$, $t = 1000$, which implementation is more efficient?

a) Optimized: $\mathcal{O}((n+t)d^2)$ approach
b) Naive: $\mathcal{O}(ndt)$ approach
c) They're equivalent

Answer: a) Optimized has better asymptotic complexity when $n + t < nd$!

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*
  - For minimization, move in direction $-\nabla f$

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*
  - For minimization, move in direction $-\nabla f$

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*
  - For minimization, move in direction $-\nabla f$
- **Algorithm Variants:**

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*
  - For minimization, move in direction $-\nabla f$
- **Algorithm Variants:**
  - Batch GD: Uses all data, smooth convergence

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*
  - For minimization, move in direction $-\nabla f$
- **Algorithm Variants:**
  - Batch GD: Uses all data, smooth convergence
  - Stochastic GD: Uses one sample, faster per iteration

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*
  - For minimization, move in direction $-\nabla f$
- **Algorithm Variants:**
  - Batch GD: Uses all data, smooth convergence
  - Stochastic GD: Uses one sample, faster per iteration

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*
  - For minimization, move in direction $-\nabla f$
- **Algorithm Variants:**
  - Batch GD: Uses all data, smooth convergence
  - Stochastic GD: Uses one sample, faster per iteration
- **Implementation Matters:**

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*
  - For minimization, move in direction $-\nabla f$
- **Algorithm Variants:**
  - Batch GD: Uses all data, smooth convergence
  - Stochastic GD: Uses one sample, faster per iteration
- **Implementation Matters:**
  - Smart precomputation: $\mathcal{O}((n+t)d^2)$

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*
  - For minimization, move in direction $-\nabla f$
- **Algorithm Variants:**
  - Batch GD: Uses all data, smooth convergence
  - Stochastic GD: Uses one sample, faster per iteration
- **Implementation Matters:**
  - Smart precomputation: $\mathcal{O}((n+t)d^2)$
  - Naive approach: $\mathcal{O}(ndt)$

Summary: Gradient Descent Optimization

- **Mathematical Foundation:**
  - Gradient ∇f points toward steepest *ascent*
  - For minimization, move in direction $-\nabla f$
- **Algorithm Variants:**
  - Batch GD: Uses all data, smooth convergence
  - Stochastic GD: Uses one sample, faster per iteration
- **Implementation Matters:**
  - Smart precomputation: $\mathcal{O}((n+t)d^2)$
  - Naive approach: $\mathcal{O}(ndt)$
  - Can be 10× speed difference!